Applications of Genetic Algorithms and Quantum Genetic Algorithms on theoretical fully-autonomous road systems

University of Birmingham

Sam Barrett
sjb786@student.bham.ac.uk

November 4, 2020

Contents

Abstract

Chapter 1

Introduction

Chapter 2

Background

2.1 Genetic Algorithms

2.1.1 History

Genetic algorithms are optimisation techniques that employ the same rationale as classical Evolution as seen in nature.

Genetic Algorithms can trace their origins back to the late 1960s when they were first proposed by John Holland, though he then referred to them as *Genetic Plans*[1]. Holland went on to write the first book on the subject titled *Adaptation in Natural and Artificial Systems*[1] in 1975. The field did not find much reception with Holland stating in the preface to the 1992 rerun:

> *"When this book was originally published I was very optimistic, envisioning extensive reviews and a kind of 'best seller' in the realm of monographs. Alas! That did not happen."*

However, in the early nineties, Genetic algorithms surged in popularity along with the area of Artificially Intelligent planning as a whole leading to Holland republishing his book and solidifying his position as the field's founder.

2.1.2 Definition

In a general sense, optimisation techniques work to find the set of parameters $\mathcal{P}$ that minimise an objective function $\mathcal{F}$. Genetic algorithms approach this

[1]This distinction was made to emphasise the *"centrality of computation in defining and implementing the plans"*[1]

by representing these sets as individuals in a population, P. Over the course of multiple generations, the best solutions are determined and promoted until termination criteria are met or the maximum number of generations is reached.

As our candidates are essentially a collection of parameters to the function we are trying to optimise, we can extend our metaphor further by mapping each element of a individual to a *gene* in a individual's genome.

The representation we use in a GA is problem specific. Often we have to provide functions to facilitate the mapping between the problem specific set of possible solutions and the encoded genotype space in which we optimise. The most basic representation being a string of binary numbers.

Genetic algorithms are both *probabilistically optimal* and *probabilistically complete*[2] meaning that: given infinite time, not only will the algorithm find *a* solution, (if one exists), it will find **the** optimal solution from the set of all possible solutions, $\mathcal{P}^*$.

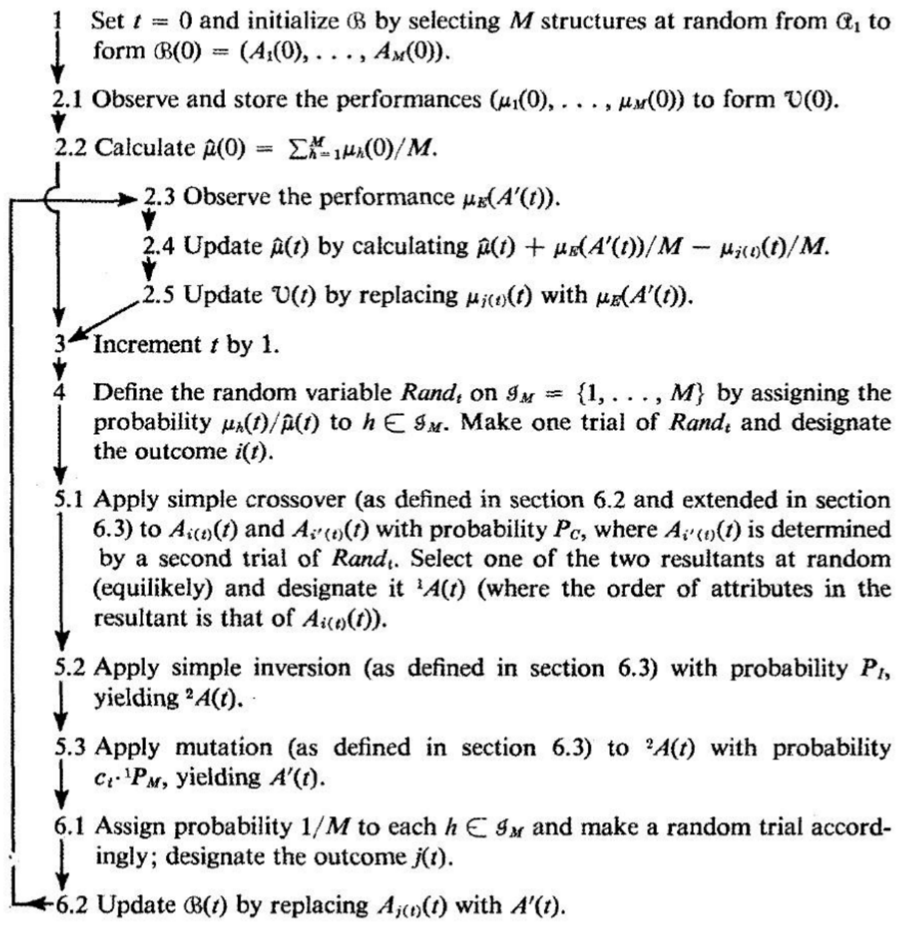
Algorithm 1: Modern Generic Genetic Algorithm

```
Result: Best Solution, $p_{\texttt{best}}$
Generate initial population, $P_0$ of size $n$;
Evaluate fitness of each individual in $P_0$, $\{F(p_{0,1}, \ldots, p_{0,n})\}$;
while termination criteria are not met do
    Selection: Select individuals from $P_t$ based on their fitness;
    Variation: Apply variation operators to parents from $P_t$ to
      produce offspring;
    Evaluation: Evaluate the fitness of the newly bred individuals;
    Reproduction: Generate a new population $P_{t+1}$ using
      individuals from $P_t$ as well as the newly bred candidates.;
    $t{+}{+}$
end
return $p_{\texttt{best}}$
```

As you can see from Figure 2.1 and Algorithm 1 the overall shape of GAs has not changed substantially over the course of the past 50 years. Being comprised of a series of operations that a starting population is piped through until criteria are met.

2.1.3 Selection

The selection procedure is the process by which the next generation of individuals is created from the current population. Individuals are selected relative to their fitness as determined by the Objective (fitness) function $\mathcal{F}$.



1 Set $t = 0$ and initialize $\mathcal{B}$ by selecting M structures at random from $\mathcal{A}_1$ to form $\mathcal{B}(0) = (A_1(0), \ldots, A_M(0))$.

2.1 Observe and store the performances $(\mu_1(0), \ldots, \mu_M(0))$ to form $\mathcal{U}(0)$.

2.2 Calculate $\hat{\mu}(0) = \sum_{h=1}^{M} \mu_h(0)/M$.

2.3 Observe the performance $\mu_E(A'(t))$.

2.4 Update $\hat{\mu}(t)$ by calculating $\hat{\mu}(t) + \mu_E(A'(t))/M - \mu_{j(t)}(t)/M$.

2.5 Update $\mathcal{U}(t)$ by replacing $\mu_{j(t)}(t)$ with $\mu_E(A'(t))$.

3 Increment t by 1.

4 Define the random variable $Rand_t$ on $\mathcal{J}_M = \{1, \ldots, M\}$ by assigning the probability $\mu_h(t)/\hat{\mu}(t)$ to $h \in \mathcal{J}_M$. Make one trial of $Rand_t$ and designate the outcome $i(t)$.

5.1 Apply simple crossover (as defined in section 6.2 and extended in section 6.3) to $A_{i(t)}(t)$ and $A_{i'(t)}(t)$ with probability P_C, where $A_{i'(t)}(t)$ is determined by a second trial of $Rand_t$. Select one of the two resultants at random (equilikely) and designate it ${}^1A(t)$ (where the order of attributes in the resultant is that of $A_{i(t)}(t)$).

5.2 Apply simple inversion (as defined in section 6.3) with probability P_I, yielding ${}^2A(t)$.

5.3 Apply mutation (as defined in section 6.3) to ${}^2A(t)$ with probability $c_t \cdot {}^1P_M$, yielding $A'(t)$.

6.1 Assign probability $1/M$ to each $h \in \mathcal{J}_M$ and make a random trial accordingly; designate the outcome $j(t)$.

6.2 Update $\mathcal{B}(t)$ by replacing $A_{j(t)}(t)$ with $A'(t)$.

Figure 2.1: GA algorithm outlined in Holland's Original Book[1]

Some methods select only the best solutions by fitness. Others employ a more stochastic approach such as the Monte Carlo method[3] to increase diversity and reduce complexity.

2.1.4 Variation

Variation in a GA is the process of altering the genome individuals to further explore the search space via stochastic local search. We perform this using two distinct sub-operations: Mutation and Crossover.

Mutation

In mutation we alter each gene with a set probability p_m known as the *mutation rate*. A standard value for a mutation rate is $\frac{1}{L}$ but it can fall anywhere in the range $p_m \in [\frac{1}{L}, \frac{1}{2}]$ where L is the length of the genome. In binary coded GAs we alter a given gene by *flipping* it's value. In real-coded GAs mutation operators include:

- Uniform Mutation
- Non-Uniform Mutation
- Gaussian Mutation

A low value for p_m a new individual which can be shown to be *close* to it's parents in the search space relative to their *Hamming distance*[2] if using a Binary coded GA. In real-coded GAs they can be shown to be close by the Euclidean distance between them.

Crossover

Crossover is a binary operation, taking two randomly selected *parents* from the population P_t with a probability $p_c \in [0, 1]$

There are two major forms of crossover for binary coded GAs: $n > 1$ point crossover and Uniform crossover.

For real-coded GAs you instead have a selection of Crossover operators including:

- Flat crossover
- Simple crossover

[2]Hamming Distance: The metric for comparing two binary data strings. The Hamming distance between two strings is the number of bit position in which they differ.

- Whole arithmetic crossover
- Local arithmetic crossover
- Single arithmetic crossover
- BLX-α crossover

2.1.5 Evaluation

After developing potential new individuals, the fitness of these new individuals is calculated using the objective function, $\mathcal{F}$

2.1.6 Reproduction

Here the next generation of individuals is constructed. There are multiple potential methods employed here with the simplest being to just replace the lest fit individuals with additional copies of the fitter individuals, be they pre-existing or newly generated.

In this stage various heuristics or alternative strategies can be implemented to speed up or slow down the convergence rate. Whilst their fitness may be low, having a diverse population allows for more of the search space to be explored. If the algorithm converges too quickly it may get *stuck* in local minima.

2.2 Autonomous Road Networks

2.3 Quantum Genetic Algorithms

2.4 Alternative Technologies

Chapter 3

Literature Review

3.1 Classical GAs

3.2 Quantum GAs

3.2.1 Quantum Computing

Chapter 4

Classical Approach

4.1 Approach

4.2 Implementation

4.2.1 Language Choice

4.3 Results

Chapter 5

Quantum Approach

5.1 Approach

5.2 Implementation

5.3 Results

Chapter 6

Evaluation

Chapter 7

Conclusion

Bibliography

[1] John H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence.* MIT Press, Cambridge, UNITED STATES, 1992.

[2] Rahul Kala. *On-Road Intelligent Vehicles: Motion Planning for Intelligent Transportation Systems / Rahul Kala.* Butterworth-Heinemann is an imprint of Elsevier, Kidlington, Oxford, UK, 2016.

[3] N Metropolis. The beginning. *Los Alamos Science*, 15:125–130, 1987.